

Neutrosophic Computing and Machine Learning , Vol. 11, 2020
University of New Mexico

Metodología para la determinación de oportunidades en la aplicación de pruebas de Software Microsoft Solutions Framework mediante números neutrosóficos

Hans Daniel Zambrano Campi[1] . Angel Braulio Martinez Vasquez[2] , Vladimir Vega Falcón PhD[3] , Silvia Patricia Chiriboga Velasco[4]

[1] Docente, Universidad Regional Autónoma de Los Andes, Ecuador. E-mail: ub.hanszambrano@uniandes.edu.ec
[2] Analista, Asociación Latinoamericana de Ciencias Neutrosóficas, Ecuador. E-mail: vasmarti10@gmail.com
[3] Analista, Investigación Uniandes Ambato, Ecuador. E-mail: ua.vladimirvega@uniandes.edu.ec
[4]Docente, Universidad Regional Autónoma de Los Andes, Sede Babahoyo, Ecuador. E-mail: ub.silviapcv.caula@uniandes.edu.ec

Resumen: Para las empresas de desarrollo de software, decidir que metodología de desarrollo utilizar al comenzar un proyecto, constituye en muchos casos una tarea importante a realizar. La selección de metodología de desarrollo puede ser modelado como un problema de toma de decisiones. En el presente artículo describe una propuesta de selección de metodología mediante números neutrosóficos a partir de lo cual se decide la propuesta de metodología de desarrollo.

Keywords: Metodologías Ágiles, MSF, Pruebas.

1 Introducción

Las metodologías ágiles son un tema ya establecido en la ingeniería de software que han suscitado interés y aceptación en su aplicación en proyectos de poca duración y con requisitos cambiantes, el esquema tradicional exige una definición rigurosa de roles, actividades y artefactos acompañado de un modelado y documentación detallada [1]. *The Agile Allianc*e, es una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil [2], [3].

Según el Manifiesto se valora al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Las personas son el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades[4], [5].

Desarrollar software que funciona, más que conseguir una buena documentación. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental. La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

Responder a los cambios, más que seguir estrictamente un plan [6], [7]. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto a metas a seguir y organización del mismo. Los principios son:

La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- El equipo del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y

confiar en ellos para conseguir finalizar el trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Tabla 1 Comparación entre metodologías ágiles y "pesadas"

Metodología ágil	Metodología tradicional
Basada en heurísticas proveniente de prácticas de producción de códigos	Basadas en normas proveniente de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente por el equipo	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas y normas
No existe contrato tradicional o al menos bastante flexible	Existe un contrato prefijado
Grupos pequeños (-10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuido
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

2 Preliminares

La sección presenta las bases teóricas para la comprensión de la propuesta de investigación. Describe *Microsoft Solutions Framework* para el desarrollo de aplicaciones ágiles. Introduce la teoría asociada sobre los probadores de software, las pruebas de validación. Finalmente, describe los números neutrosóficos en el contexto de la presente investigación para la selección de la metodología.

2.1 Microsoft Solutions Framework

Microsoft Solutions Framework (MSF) para el Desarrollo de Aplicaciones Ágiles, es la propuesta de Microsoft para el desarrollo de aplicaciones usando una metodología ágil; la cual incorpora prácticas para manipular los requerimientos de calidad de servicio (QoS), tales como rendimiento y seguridad [8], [9].

Las fases del MSF son las siguientes:

Fase 1 - Estrategia y alcance

Fase 2 - Planificación y Prueba de Concepto

Fase 3 - Estabilización

Fase 4 – Despliegue

Los roles en MSF Ágil

En el modelo de equipo de MSF no hay jerarquías, todos son igual de importantes.

Roles:

- Analista de negocios
- Jefe de proyecto
- Arquitecto
- Desarrollador
- Personal de pruebas
- Personal de despliegue
- Usuarios experimentados

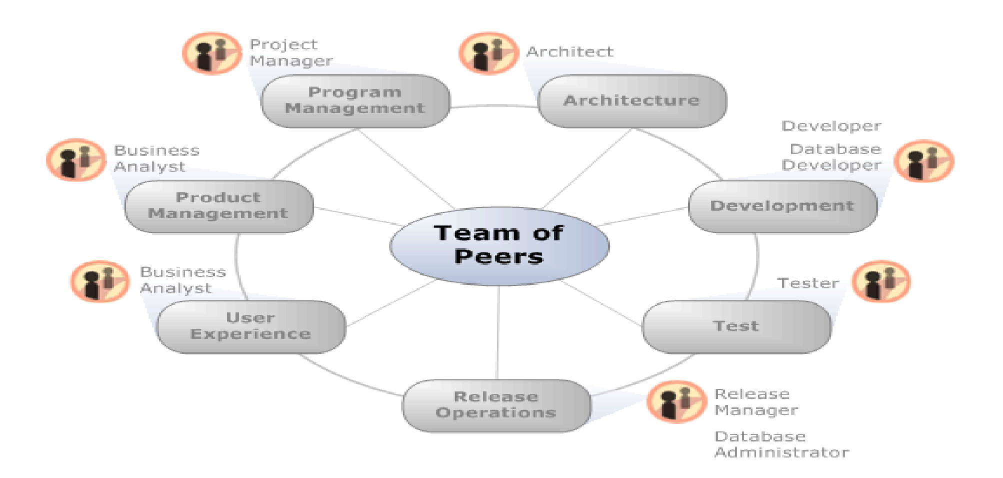


Figura 1: Roles en MSF para el Desarrollo de Aplicaciones Ágiles

Este equipo de especialistas, con roles diferentes, se reúne en representación de todos los miembros implicados con la producción, uso y mantenimiento del producto. Cada integrante del equipo, o cada rol, es responsable de representar las necesidades específicas de todos los miembros de su grupo y ninguno es más importante que otro. Estas reuniones proporcionan los balances y chequeos necesarios para garantizar que se produzca una verdadera solución. La figura 1 muestra la composición del equipo de especialistas.

2.2 Probadores de Software

El personal de pruebas se encarga del área de pruebas en MSF *Team Model*. Su objetivo principal es encontrar y comunicar los problemas del producto que podrían afectar negativamente a su valor. El personal de pruebas debe comprender el contexto del proyecto y ayudar a otras personas a basar sus decisiones en dicho contexto. Un objetivo clave del personal de pruebas es localizar los errores significativos que presenta el producto durante la fase de pruebas e informar consecuentemente. Una vez que se encuentra un error, también corresponde al personal de pruebas comunicar con exactitud sus consecuencias y describir las soluciones temporales que reducirían su impacto. Deben redactar descripciones de los errores y de los pasos necesarios para reproducirlos, fáciles de comprender y de seguir. También participan con el resto del equipo en la definición de los estándares de calidad para el producto. El propósito de las pruebas es comprobar si las funciones conocidas desempeñan su labor correctamente y descubrir nuevos problemas.

El flujo de trabajo del personal de pruebas es el siguiente:

- Análisis
- Cerrar errores
- Desarrollo de la documentación
- Establecimiento de los entornos de prueba
- Establecimiento del proceso del proyecto
- Lanzamiento del producto
- Prueba de un requisito del cliente
- Comprobación de un requisito del producto

2.3 Probadores de escenarios

Un escenario es dividido en tareas de prueba y tareas de desarrollo y los probadores son los encargados de desarrollar y ejecutar los casos de prueba. La asignación de un escenario para poner a prueba, indica que la funcionalidad ha sido integrada a una estructura y está lista para ser probada [10], [11]. Validar que una estructura refleja la funcionalidad prevista en el escenario, requiere una comprensión del escenario y sus condiciones del límite; por lo que las pruebas de validación deben escribirse de forma que cubran la funcionalidad completa, así como la condición de límite del escenario y serán ejecutadas mientras se reporten *bugs*

¿Qué hacer para probar escenarios?

MSF plantea un conjunto de actividades que se deben realizar para probar escenarios:

Definir aproximación de prueba

Una aproximación de prueba es una estrategia que guía el plan de prueba y su ejecución, a la vez que determina los modelos de calidad para empaquetar el producto [12], [13]. Las actividades de aproximación de prueba son un punto de arranque para el plan de prueba anticipado del proyecto, pero evoluciona y cambia con este. Una aproximación de prueba debe incluir una mezcla de técnicas, incluyendo el manual y las pruebas automatizadas y antes de cada iteración, el documento de aproximación de prueba debe ponerse al día para reflejar las metas de la comprobación de la iteración y los datos de prueba que serán empleados.

Las sub-actividades definidas para esta actividad son:

- Determinar el contexto del proyecto: Se identifican los riesgos del proyecto y los usuarios que estos podrían afectar, así como las situaciones especiales que podrían impactar el nivel de comprobación necesario. Debe determinarse lo que está en riesgo y su impacto, en caso de que el producto falle.
- Determinar la misión de la prueba: Se identifican las metas del proyecto a ser satisfechas a través de las pruebas, consultando con el arquitecto y analista de negocio en las incertidumbres técnicas y riesgos del usuario.
- Evaluar posibles técnicas de prueba: Se deben evaluar las herramientas disponibles para realizar pruebas, tanto como las habilidades del equipo de prueba, a fin de determinar las técnicas de pruebas posibles y apropiadas para el proyecto.
- Definir métricas de la prueba: Se debe usar el contexto del proyecto, misión de la prueba, y técnicas de prueba para determinar las métricas de prueba. Estas métricas incluirán los umbrales para los varios tipos de pruebas (carga, rendimiento, etcétera) o el porcentaje de pruebas automatizadas.

2.4 Pruebas de validación

Las pruebas de validación aseguran la funcionalidad del sistema, toman una vista de “caja negra” de la aplicación y se concentra en las áreas más importante para el usuario final con el objetivo de chequear la funcionalidad se corresponde con lo escrito en el escenario [14], [15]. Escribir los casos de pruebas para las pruebas de validación ayuda a los probadores a identificar problemas usando mecanismos de prueba que imitan el mundo real.

Para escribir una prueba de validación debe tener en cuenta los siguientes aspectos:

- Identificar el área y ambiente de prueba: Aísle el área dónde se correrá la prueba. Las pruebas de iteración son el conjunto de casos de prueba automatizados que corren después de las pruebas de validación de funcionalidad. Sin embargo, una funcionalidad no tiene que pasar esta prueba para ser exitosa. Las pruebas pueden correrse como parte de las pruebas de iteración si son automatizadas.
- Identificar los detalles del flujo de los casos de prueba: Identifique los datos de prueba requeridos para cada caso de prueba, apoyándose en el informe de aproximación de prueba, así como, las restricciones y condiciones de límite para los casos de pruebas requeridos en las tareas de prueba. Chequee si los casos de prueba pueden automatizarse e identifique los pasos procesales para el flujo del escenario.
- Escribir casos de prueba: Escriba la documentación de las pruebas para los manuales de casos de prueba y los casos de pruebas automatizadas para las pruebas de iteración.
- Otras actividades a ejecutar son: la selección de un caso de prueba para correrlo, descubrir posible bug y realizar pruebas exploratorias se refiere en el punto dedicado a las pruebas de los requerimientos de calidad de servicio.

2.5 Números neutrosóficos para la determinación de oportunidades en la aplicación de pruebas

La determinación de oportunidades en la aplicación de pruebas software puede ser modelado como un problema de toma de decisión multicriterio [16, 17]. A partir del cual se poseen un conjunto de alternativas que representan las pruebas de desarrollo de software $\mathrm{A} = \{\mathrm{A}_1, \dots \mathrm{A}_\mathrm{n}\},\ \mathrm{n} \geq 2$; las cuales se les realizan una valoración a partir del conjunto de criterios $\mathrm{C} = \{\mathrm{C}_1, \dots \mathrm{C}_\mathrm{m}\},\ \mathrm{m} \geq 2$ que caracterizan el proyecto.

La solución está definida con una espectro que representa la preferencia verdadera de usar un tipo de prueba con un grado de falsedad y un grado de negación. Problema de esta naturaleza han sido modelados como un problema neutrosófico.

La neutrosofía permite la representación de la neutralidad, fue propuesta por Smarandache [18]. Representa las bases para una serie de teorías matemáticas que generalizan las teorías clásicas y difusas tales como los conjuntos neutrosóficos y la lógica neutrosófica. Un número neutrosófico (N) se representa de la siguiente forma [19]: sean $N = \{(T, I, F) : T, I, F \subseteq [0, 1]\}n$, una valuación neutrosófica es un mapeo de un grupo de fórmulas proporcionales a N , esto es que por cada sentencia p se tiene:

$$v\,(p) = (T, I, F) \tag{1}$$

Donde:

T: representa la dimensión del espacio que representa la verdad,

I: representa la falsedad,

F: representa la indeterminación.

Una valuación neutrosófica es un mapeo de un grupo de fórmulas proporcionales a N , donde por cada sentencia p se tiene:

$$v\ (p)\ =\ (T, I, F)$$

3 Implementación de pruebas sobre la calidad del servicio

Para validar que una estructura refleja las restricciones previstas en los requerimientos de calidad de servicio, se necesita conocimiento más allá de la restricción y que las pruebas de rendimiento, seguridad, esfuerzo y carga sean completadas y ninguna sea bloqueada. MSF plantea un conjunto de actividades que se deben realizar para probar los requerimientos de calidad de servicio: La aproximación de prueba ya fue descrita cuando tratamos las pruebas de escenarios, en la que se decía que es el paso que precede a la creación del plan de pruebas. El plan de pruebas permite especificar lo que desea probar y cómo ejecutar y medir el progreso de dichas pruebas [20]. La figura muestra 2 muestra las actividades fundamentales durante la planificación del plan de pruebas.

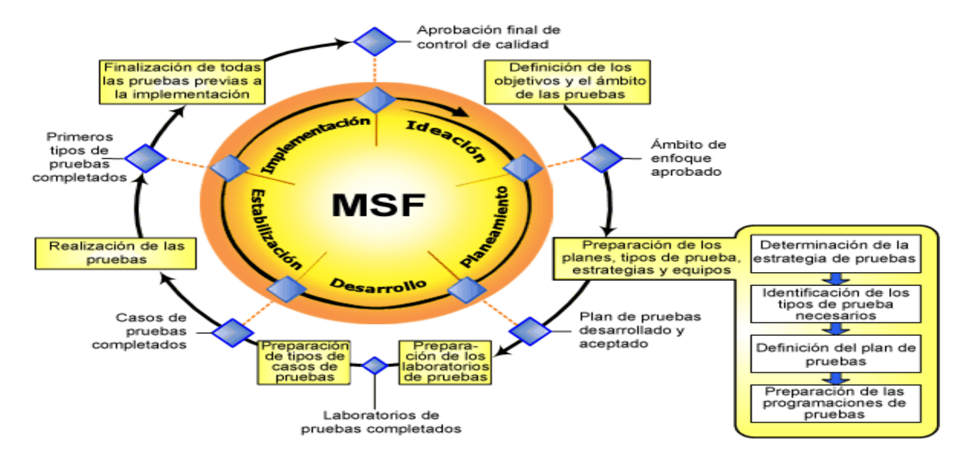


Figura 1: Actividades durante la etapa de pruebas.

Para definir una prueba de rendimiento hay que realizar las siguientes sub-actividades:
Entender el objetivo de la prueba
Especificar la configuración de la prueba: Las variables de configuración incluyen el hardware, sistema operativo, software y otras características cuyo uso es importante para ejecutar las pruebas. Cada configuración de pruebas puede representar una entrada de la matriz de pruebas. Ver figura 3.

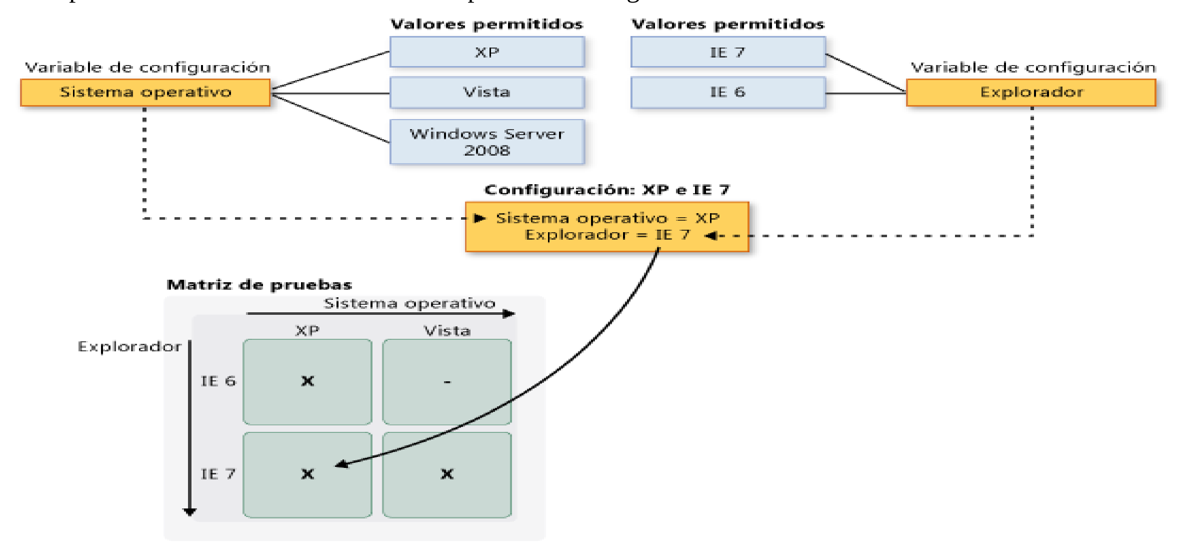


Figura 2: Variables de configuración de pruebas.

Diseñar la prueba: Se deben mapear las condiciones de prueba incluidos los requisitos previos y el escenario programado que debe ser revisado para determinar en qué áreas pueden ser crítico el rendimiento.

Las pruebas de seguridad o pruebas de penetración emplean las amenazas encontradas en el proceso de modelado de amenazas para simular un intento del adversario para atacar el producto. Esta forma de prueba se puede dividir en tres partes: exploración, identificación del defecto y la explotación. Las pruebas de penetración pueden descubrir nuevas vulnerabilidades que se convierten en requisitos de seguridad o errores en el intento de bloquear los puntos de entrada y el posterior acceso a los activos.

Esta forma de prueba requiere habilidades especiales de poder pensar y actuar como el adversario. Las pruebas de esfuerzo determinan puntos de rotura de una aplicación y pone la aplicación más allá de su límite superior en que los recursos están saturados. Se utilizan para identificar los límites superiores de carga de la aplicación donde la respuesta de aplicación se ha degradado a un nivel inaceptable o ha fracasado completamente.

Una prueba de esfuerzo es un tipo de prueba de rendimiento. Pueden ser utilizadas para predecir comportamiento de las aplicaciones y para validar la estabilidad de una aplicación y la fiabilidad mediante la ejecución de las pruebas de carga durante un período prolongado de tiempo.

Una prueba de carga es otro tipo de prueba de rendimiento. Las pruebas de carga ayudan a asegurar que la solicitud cumple con sus requisitos de calidad de servicio bajo condiciones de carga. Al ejecutar pruebas de carga, es común que se centre en las áreas de alto tráfico, el 20% de la aplicación que se utiliza el 80% del tiempo.

El testing exploratorio es una forma sistemática de realizar pruebas de un producto el objetivo es descubrir nuevos escenarios o nuevos requerimientos de calidad de servicio. Es importante definir un rango de tiempo límite para la prueba y llevar una bitácora.

Otras dos actividades a realizar por los probadores son: Seleccionar y ejecutar un caso de prueba y descubrir un bug.

7 Principales resultados

La presente sección describe un ejemplo para demostrar la aplicabilidad del método propuesto en un caso de selección del tipo de prueba. El ejemplo presenta los elementos fundamentales sintetizados para facilitar la comprensión de los lectores. Los principales criterios valorativos que se tuvieron en cuenta para la selección de la prueba estuvieron compuestos por los siguientes 5 indicadores:

c_1 Redundancia,
c_2 Complejidad,
c_3 Dinamismo,
c_4 Especialización,
c_5 Personal.

Paso 4. Determinación los pesos de los criterios evaluativos.

A partir de la consulta realizada a expertos se obtuvieron los vectores de importancia W atribuidos a cada indicador. La Tabla 1 muestra los valores resultantes de la actividad.

Tabla 1: Pesos determinado para los indicadores.

Indicadores	**Pesos W**
1	0.63
2	0.85
3	0.74
4	0.88
5	0.94

Se realiza un procesamiento de las evaluaciones sobre el cumplimiento de los criterios.

A partir de las evaluaciones expresadas por los expertos sobre el comportamiento de los indicadores en el caso de estudio se obtienen las preferencias promediadas por indicadores tal como expresa la Tabla 2.

Tabla 2: Resultado de las preferencias.

Criterios	**C1**	**C2**	**C3**	**C4**	**C5**
Evaluación	MD	M	MA	B	B

A partir del resultado de las preferencias se obtuvo un vector de preferencia tal como se expresa:

S=[0.80, 0.7, 0.84, 0.75, 0.66]

Finalmente, para el caso de estudio se obtuvo una evaluación general: $\mathrm{E} = 0.75$

El resultado expresa que la recomendación se produce hacia la utilización de pruebas dinámicas.

A partir de la selección de la prueba dinámica, se inicia el proceso de prueba.

Para el caso específico de las pruebas se puede decir que MSF Ágil concede gran importancia a las pruebas y permite un aprovechamiento de las herramientas integradas al Visual Studio (VS), aunque pueden emplearse otras herramientas.

Las pruebas pueden ejecutarse dentro y fuera del VS.Net .

Internamente: Test Manager y Test Results

Externamente: MSBuild (script)

VS Team Suite permite la creación de Work Items (tasks y/o bugs) asociados a la ejecución de las pruebas

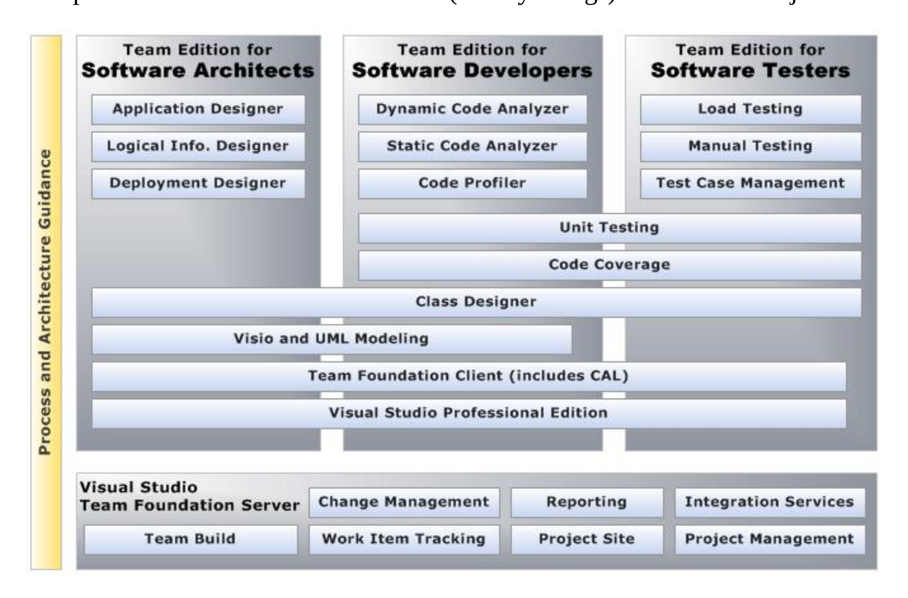


Figura 4: Visual Studio Team Suite como herramienta de soporte

VSTS dispone de funcionalidades para la realización de pruebas de aplicaciones Web. Desde Visual Studio se puede realizar la grabación de una navegación para, posteriormente, añadirle reglas que validen las respuestas. También dispone de capacidades para la generación de pruebas de carga mediante la definición de escenarios.

Conclusiones

La determinación de oportunidades en la aplicación de pruebas de software representa una tarea importante al inicial proceso de desarrollo. Este conocimiento constituye un problema de decisión discreto que puede ser modelado mediante números neutrosóficos.

Microsoft Solutions Framework representa una metodología ágil que provee una vinculación más estrecha con los clientes y busca que todos los productos sean entregables. Es una metodología flexible, de fácil manejo y que tiende a simplificar la gestión del proyecto para aplicaciones pequeñas y a corto plazo.

Referencias

[1] A. N. Cadavid, J. D. F. Martínez, and J. M. Vélez, “Revisión de metodologías ágiles para el desarrollo de software,” *Prospectiva,* vol. 11, no. 2, pp. 30-39, 2013.

[2] J. H. Canós, and M. C. P. P. Letelier, “Metodologías ágiles en el desarrollo de software,” 2012.

[3] P. Letelier, and M. C. Penadés, “Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP),” 2006.

[4] O. T. Gómez, P. P. R. López, and J. S. Bacalla, “Criterios de selección de metodologías de desarrollo de software,” *Industrial data,* vol. 13, no. 2, pp. 70-74, 2010.

[5] A. O. Duarte, and M. Rojas, “Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo,” *Revista Avances en Sistemas e Informática,* vol. 5, no. 2, pp. 159-171, 2008.

[6] M. Figueroa, “MeISE: Metodología de ingeniería de software educativo,” *Revista Internacional Internacional Internacional de Educación en Ingeniería Educación en Ingeniería ISSN,* vol. 1116, 1940.

[7] M. Arias, Á. López, and J. Honmy, “Metodología dinámica para el desarrollo de software educativo,” 2015.

[8] S. Machiraju, and R. Modi, “Developing Bots with Microsoft Bots Framework,” 2018.

[9] G. O. Tashchiyan, A. V. Sushko, and S. V. Grichin, "Microsoft Business Solutions-Axapta as a basis for automated monitoring of high technology products competitiveness." p. 012065.

[10] C. Tascon, and H. Domínguez, “Análisis a la utilidad de la técnica de escenarios en la elicitación de requisitos,” *Revista Antioqueña de las Ciencias Computacionales*, vol. 7, no. 1, 2017.

[11] J. V. Jimeno Flores, C. Hernández, and G. Milckar, “Metodología para medir el nivel de rendimiento de los probadores de software en la empresa G & V Servigen SAC,” 2018.

[12] E. Serna, R. Martínez, P. Tamayo, and I. U. de Envigado, “Una revisión a la realidad de la automatización de las pruebas del software,” *Computación y Sistemas*, vol. 23, no. 1, pp. 169-183, 2019.

[13] C. C. Villada Zapata, C. A. Cifuentes Hincapie, V. M. Delgado Pena, and O. H. Rojas García, “Profundización de pruebas de software website MercadoLibre,” 2019.

[14] O. Mar, and B. Bron, “Base Orientadora de la Acción para el desarrollo de prácticas en un Sistema de Laboratorios a Distancia ” *Revista Científica*, vol. 2, no. 29, pp. 140-148, 2017.

[15] A. Dávila, C. García, and S. Cóndor, “Análisis exploratorio en la adopción de prácticas de pruebas de software de la ISO/IEC 29119-2 en organizaciones de Lima, Perú,” *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*, no. 21, pp. 1-17, 2017.

[16] A. Grajales Quintero, E. Serrano Moya, and C. Hahan Von, “Los métodos y procesos multicriterio para la evaluación,” *Luna Azul*, vol. 36, no. 1, pp. 285-306, 2013.

[17] C. Bouza. "Métodos cuantitativos para la toma de decisiones en contabilidad, administración, economía," https://www.researchgate.net/publication/303551295_METODOS_CUANTITATIVOS_PARA_LA_TOMA_DE_DECISIONES_EN_CONTABILIDAD_ADMINISTRACION_ECONOMIA.

[18] F. Smarandache, “A Unifying Field in Logics: Neutrosophic Logic,” *Philosophy*, pp. 1-141, 1999.

[19] H. Wang, F. Smarandache, R. Sunderraman, and Y. Q. Zhang, *Interval Neutrosophic Sets and Logic: Theory and Applications in Computing: Theory and Applications in Computing*: Hexis, 2005.

[20] E. Vázquez-Cano, “Dificultades del profesorado para planificar, coordinar y evaluar competencias claves. Un análisis desde la Inspección de Educación/Teachers' difficulties to plan, coordinate and evaluate key competencies. An analysis from the education inspection,” *Revista Complutense de Educación*, vol. 27, no. 3, pp. 1061-1083, 2016.

Received: octubre 28, 2019. Accepted: enero 16, 2020